Network simulator – Object oriented programming with C++ 2021

Introduction and purpose

Communication networks exist as part of an integral part of everyday human to human interaction. They serve as the basis of interconnection for the purpose of gathering, processing, and distributing information amongst computers. The scope of computers as used here includes devices (Nodes) such as servers, routers, workstation, wireless extension points, and base stations with the interconnection media taking the physical form of copper cables, fiber optics, and microwave satellite and radio links.

Owing to the complexity of current communication networks with several layers of networks embedded or connected to other networks, it is overly complex for traditional analytical methods to provide accurate understanding of system behaviors. It is here network simulators find they are useful. In simulators, the computer network is modelled with Nodes, Links, and applications out of which performance analytics or summary is reported. The nodes characteristic feature involves forwarding and/or receiving packets intended for other nodes. The link is ascribed to the functionality of providing a media through which packets are transmitted and is characterized by the transmission speed, which is the interval of packet transmission, and the delay propagation, which is the time interval it takes for packets to travel across the link. And lastly, an application which either sends or receives packets.

A computer network simulation can be thought of as a flow of interaction among network Nodes through transmission of packets. These packets move through the system, interact with other entities, join activities, trigger events, cause some changes to state of the system, and terminate themselves. Occasionally they contend or wait for some type of resources implying that they are executed in a logical and sequential manner. The packets themselves are embedded with the destination IP address, the data to be transmitted and error detection or sequential information.

Scope of the work

This project will feature a GUI driven simulator built on top of the Qt framework. With this simulator, users can build different network scenarios by adding nodes and links between them and then further configure the dataflow between different nodes. Figure 1 demonstrates the idea of having different nodes and links with different properties such as IP addresses, transmission speed, and queue sizes.

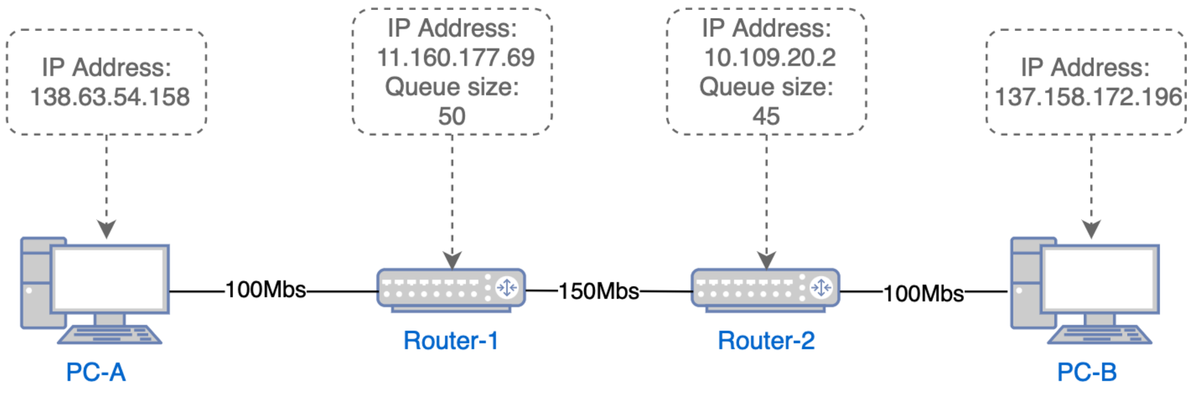


Figure 1. Different nodes in the network simulator

Once user have added the needed nodes to the system, different dataflow scenarios can be configured. The configuration includes setting the source and destination nodes and data size. Figure 2 demonstrates the chain of events that take place when running the simulator. Once the configurations are done, the simulator will compute the different scenarios and progress will be displayed to the user. When the computation is finished, the simulator will display performance statistics of different scenarios.

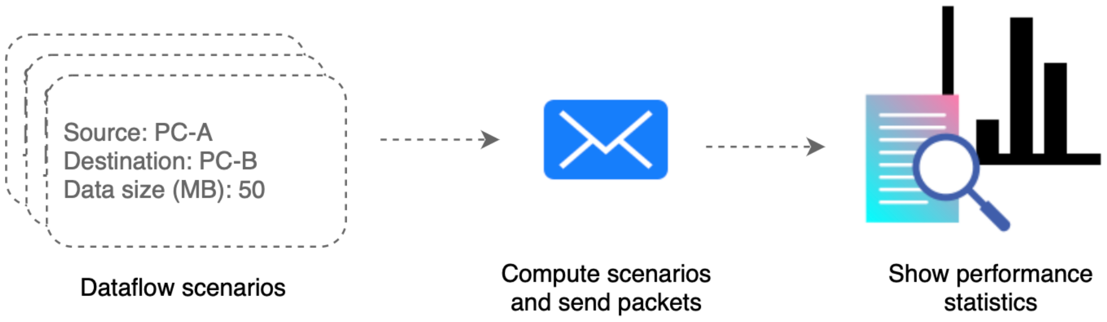


Figure 2. Simulator's chain of events

Functionalities and features

Adding nodes

The simulator includes two different nodes: *routers* and *end-hosts*. Routers can only forward data and end-hosts can send and/or receive data. User can place the nodes wherever wanted in the given space and name them. The routers also enables setting a queue size.

Adding links

Once user have added at least two nodes, the user can add link between them. When adding a link, user need to set the nodes and also the link's transmission speed.

Configuring packet flow scenario

One simulator event can have one or more scenarios of dataflows. Configuring a scenario, user needs to set the source node that will send the data, the destination node that will receive the data, and the size of the data that is to be sent.

Follow the progress of simulation

When the simulator is computing the scenarios, the progress will be either displayed visually or printed out in text form.

Requirements

In this session, we will describe the key elements that will enable efficient features in the Network Simulator. There are implementations of different entities (Hosts, Links, Packets, packets flow, Routers, and Switches) that represent the network objects.

The blueprints (classes) of these network objects and their relationships should be implemented, having all their relevant members and methods. These objects will hold configurations and interact with one another to trigger, update, or remove network events and different states of the system.

As required in a typical computer network, unique IP addresses should be allocated to each of the End-hosts and routers so that nodes are easily identified in the network and packets are routed on best paths to their destinations.

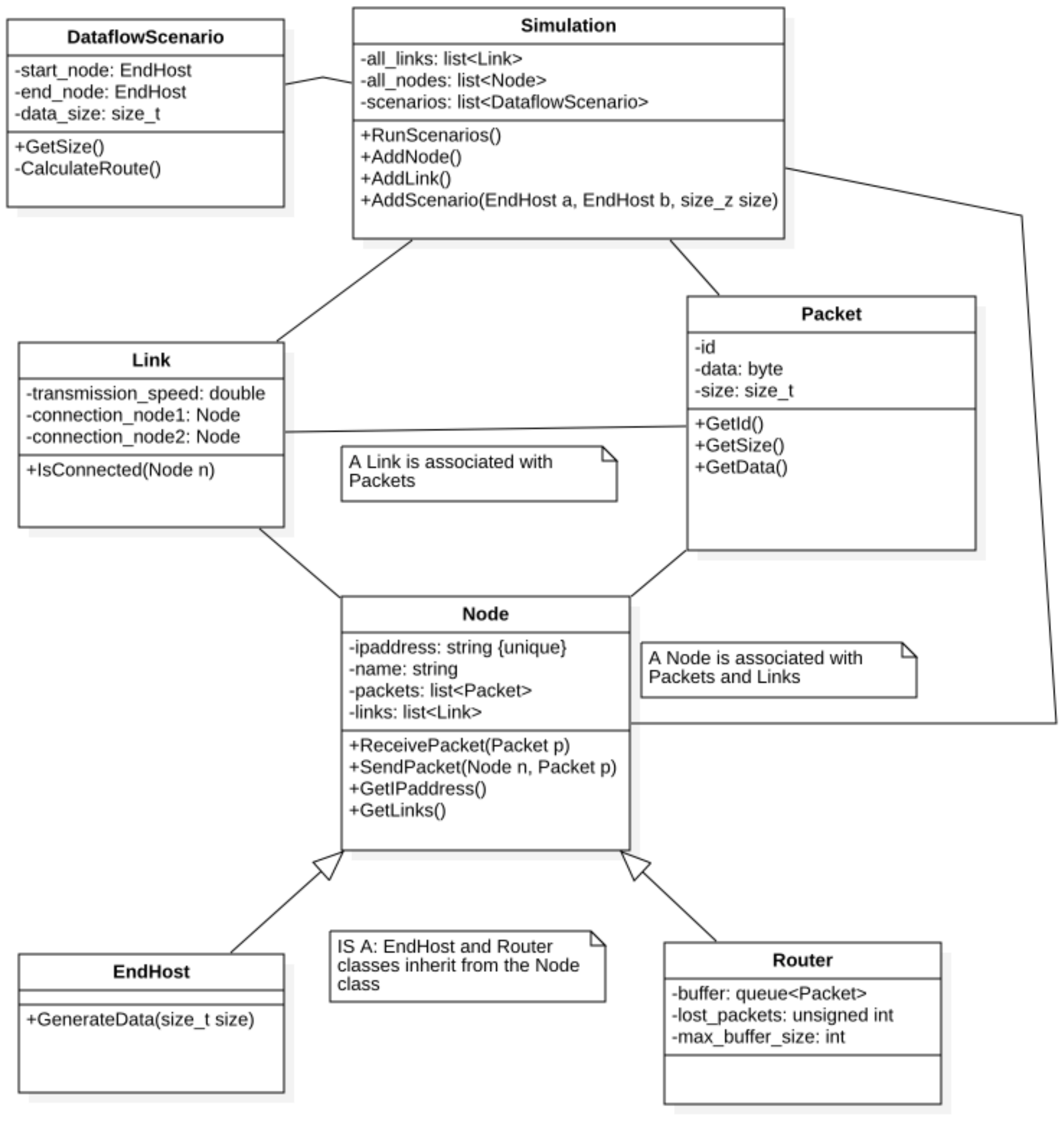
Implementations of Router's functionalities: receives and forwards packets to their destination should be considered in correlation to End-hosts that send and receive packets.

The Link objects should be implemented to manage packet transmissions. Transmission speed, propagation delays, and queue should be considered members of the Link object.

The concept of TCP/IP should be implemented at a bare minimum (at least each node in the network is identified by its IP address) to support communication between these nodes.

A GUI with Qt will be implemented to give users visual representation of the network events and handy buttons that let them trigger and have control of these network events. The Qt functionality should be combined with C++ to provide a suitable graphic user interface for the network simulation.

High-Level structure



Division of work

The team has divided the group roughly into two subgroups: Front-end and back-end development. However, the team will work together building the software and will be held responsible as a whole. Communication between members is done via Telegram and Teams.

The group division:

Prince Okumo & Milla Sipilä – GUI

Oriyomi Oladele & Hechukwedere Okoreogo – Backend

Planned schedule

Week	Dates	Tasks
44	1.11-7.11 Project plan DL!	- Make a project plan and divide work - Discuss coding and git practices
45	8.11-14.11	- Start building the code base - Get familiar with tools and libraries
46	15.11-21.11	- The most basic implementation done (POC) -> add nodes and send data between them - Test end-to-end product with the team
47	22.11-28.11	- Fix problems that were encountered in the testing - Add features and complexity
48	29.11-5.12	- Refine GUI if there is time - Add “nice to have” features if no major problems
49	6.12-12.12	- Testing and bug fixing, no new features